Refine Search

Search Results -

Terms	Documents
L9 and L1	10

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L10

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Thursday, June 17, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L10	L9 and l1	10	L10
L9	4811345[uref]	35	L9
L8	L7 same l6	2	L8
L7	suspend$ or halt$	391312	L7
L6	L5 same interface	186	L6
L5	L4 same l1	864	L5
L4	(host adj1 computer) or tester	77605	L4
L3	L2 same interface	178	L3
L2	host same l1	472	L2
L1	test$ near4 $processor	10227	L1

END OF SEARCH HISTORY

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L8: Entry 2 of 2 File: USPT Mar 7, 1989

DOCUMENT-IDENTIFIER: US 4811345 A
TITLE: Methods and apparatus for providing a user oriented microprocessor test interface for a complex, single chip, general purpose central processing unit

Abstract Text (1):
Methods and apparatus are disclosed that facilitate the testing and development of computer systems that include at least one single chip microprocessor. In particular, a parallel test interface is described that allows an external test unit to (1) directly load instructions into the microprocessor under test utilizing the existing bus structure of the computer system; (2) step the processor through preselected test instruction sequences; (3) monitor processor states in both the processor's test and normal execution modes; and (4) halt and resume normal instruction processing. According to the invention, the microprocessor test interface comprises a plurality of dedicated CPU status output pins and a plurality of dedicated CPU control input pins, used by the test unit in combination with the existing bus structure of the computer system to provide the desired test facility for the single chip microprocessor. The preferred embodiment of the invention is realized in a RISC environment where the instruction lengths are fixed and the instruction processor has a single cycle execution time. Such an embodiment facilitates the direct insertion of instructions by the tester into the processor for decoding, without having to queue instructions or pass through complicated intervening hardware or test logic.

Detailed Description Text (65):
The novel test interface described hereinbefore is asynchronous to the processor clock input and thus may be driven directly by an external processor, such as a tester or hardware development system, directly under the control of software in the external processor. The interface allows the external processor to place processor 130 in the HALT mode, to trace the execution of instructions on an instruction-by-instruction basis, and to inspect and alter the state of processor 130 and external devices and memories.

L10: Entry 6 of 10 File: USPT Oct 25, 1994

DOCUMENT-IDENTIFIER: US 5359547 A
TITLE: Method and apparatus for testing processor-based computer modules

Abstract Text (1):
A method and apparatus for testing complex processor-based computer modules and their associated computer systems by allowing the normal initialization path between a memory component storing code utilized during initialization and the processor to be interrupted and test code from an external test system to be substituted for initialization code. Following initialization, a two-way communication link between the processor and the test system is created to allow interactive testing and status reporting. The testing method and apparatus maximizes the likelihood of precisely identifying defects on the module under test.

Brief Summary Text (2):
The present invention relates generally to computer module testing systems, and more particularly to a computer module having a processor and formed to enable dynamic testing of the module by an external test system using test sequences which can be selectively introduced to the processor.

Brief Summary Text (6):
A common technique for testing complex modules with on-board processors involves the addition of test sequences to the processor's initialization code. Many processor-based computer modules have Read-Only Memory (ROM) components which supply code utilized by the processor during initialization. The inclusion of test sequences with the initialization code stored in the ROM allows the processor to initialize and test itself, and begin testing other components on the module. The code utilized during initialization may also contain a pointer to additional code stored in other memory components on the MUT. The additional code stored in these memory components comprises more extensive test sequences.

Brief Summary Text (9):
Once the test sequences are retrieved and executed by the processor, status reporting is usually accomplished through some general manifestation of a pass or fail condition on the module itself. For example, a light emitting diode (LED) may be activated by the processor if the module passes all the tests within the test sequence or remain deactivated if the module fails any tests. If a failure is indicated in such a way, there may be no information available as to which of multiple tests within a testing sequence the module failed or what the defect might be.

Brief Summary Text (14):
Another testing method which has been used for complex modules replaces the ROM which stores the code utilized by the processor during initialization with a plug-type connector attached to the MUT where the ROM pins would normally be attached. The plug-type connector allows attachment to test equipment through a cable. The test equipment can provide the processor with code which includes initialization code as well as test sequences to be executed by the processor. This allows new test sequences to be loaded into the processor by re-initializing the processor instead of removing the ROM and blasting new code into the ROM each time new

sequences are needed.

Brief Summary Text (15):
Many of the difficulties associated with testing methods which include initialization code and test sequences in the ROM itself may also be experienced when using this ROM replacement testing method. As previous pointed out, if the processor needs to rely on a significant amount of untested hardware to access the ROM, this ROM replacement testing method may not be particularly useful. Also, as previously mentioned, the amount of code accepted by the processor during initialization may be limited which may prevent the loading of comprehensive tests. Incremental loading of comprehensive tests may also not be possible through this ROM replacement testing method, because for additional test sequences to be loaded, the processor needs to be re-initialized, which would erase any previously loaded code. Additionally, as pointed out above, status reporting may be limited to manifestations on the MUT. A pass/fail indication provides no information about which test the module failed or what type of defect is present.

Brief Summary Text (21):
The foregoing and other objects of the present invention may be accomplished by providing a testing method and arrangement which includes on a module under test (MUT) interruption hardware disposed between the memory component used to store code utilized during initialization (hereinafter referred to as initialization code) and the processor which permits interruption of the normal initialization path to the processor, and means for substituting a test code for the initialization code. The testing arrangement also includes hardware which allows the MUT to be connected to an external testing system such that the external testing system may substitute the test code for the initialization code during processor initialization and receive corresponding responses.

Brief Summary Text (24):
Following initialization, a two-way communication link is established between the MUT and the test system through the testing arrangement. This two-way communication link between the processor and the test system allows interactive test loading and status reporting without having to re-initialize the processor. The amount of test code which can be loaded interactively is not limited to the amount of code accepted by the processor during initialization.

Drawing Description Text (2):
FIG. 1 is a block diagram depicting a testing arrangement in accordance with the invention for testing a processor-based computer module using an external testing system; and

Detailed Description Text (2):
The invention generally involves a method and apparatus for testing complex modules, such as processor-based computer modules and will be described in that context. It will be appreciated, however, that the invention has applicability to testing other types of modules and to other testing situations.

Detailed Description Text (7):
Also included in the testing arrangement, as shown in FIG. 1, a multiplexor (MUX) 36 or other logic switching element may be disposed between the processor 24 and the ROM 26, such that the initialization line 32 is attached to a first input of the MUX and the data input line 34 is attached to an output of the MUX. As will be described, the MUX 36 functions as a means for allowing the initialization path between the ROM and the processor (initialization line 32 / data input line 34) to be interrupted and permits the substitution of test code from the test system 22 for the initialization code from the ROM. The test code may comprise processor initialization code, a testing sequence, or controller code, which will be discussed in more detail hereinafter.

Detailed Description Text (9):
The test system 22 is electrically connected through the test connector 38 to a second data input of the MUX 36, referred to hereinafter as the test code input line 44, to allow test code to be supplied to the MUX. The test system is further electrically connected through the test connector 38 to a select line 46 of the MUX to allow the test system to control the flow through of the MUX. The test system may provide test code to the processor through the test code input line 44 by asserting the select line 46 of the MUX 36 to cause the MUX to permit data on the test code input line to flow through the MUX to its output in place of the initialization code from the ROM 26 on initialization line 32. The MUX'S output is directly connected to the processor's data input line 34 used to provide the initialization code to the processor. The MUX thus serves as a switch that interrupts the initialization path between the ROM and the processor to enable external test code from test system 22 to be substituted for the initialization code from the ROM.

Detailed Description Text (10):
The test system 22 is also electrically connected through the test connector 38 to the processor's reset line 28 and data strobe line 30. In order for the test system to load the processor with test code as a substitution for the initialization code, the test system first interrupts the normal ROM to processor initialization path as described above. It then asserts the reset line 28 to the processor to begin initialization, and next deasserts the reset line and provides data through the MUX 36 in response to the processor's request for data. As described previously, the processor requests data through the use of the data strobe line 30.

Detailed Description Text (11):
As will be appreciated by those skilled in the art, the MUX 36 is only one embodiment of many possible interruption devices which may be used to substitute test code for the initialization code. Any switch or other functionally equivalent combination of module-resident hardware (combinational logic) will suffice. However, since untested logic is being relied upon to load the test code into the processor 24, the number of connections should be kept to a minimum to reduce the possibility of defects that might have an adverse effect upon code loading. In the preferred embodiment of the invention, the initialization path is a serial path (as shown in FIGS. 1 and 2). A serial path is preferred over a parallel path due to the fact that fewer connections need to be defect-free to initialize the module. However, a parallel path between the processor 24 and ROM 26 also may be used with a parallel interruption means, i.e., a MUX 36 with parallel paths or functionally equivalent combinational logic.

Detailed Description Text (12):
By interrupting the path between the ROM 26 and the processor 24 a relatively unintrusive testing method is created which does not alter the operating functionality of the MUT 20. This reduces the possibility that defects will be created or that existing defects will be masked by the testing.

Detailed Description Text (15):
Following initialization, the processor may execute the code located in its I-cache. The test code may include processor initialization code as well as a testing sequence to be executed by the processor. The testing sequence can include whatever tests are deemed necessary, and is limited only by the size of the I-cache during initialization.

Detailed Description Text (16):
The test code may also include controller code which will allow the processor 24 to establish a two-way communication link with the test system 22 after initialization. The controller code can bring the processor up and configure the processor so that an input signal from the test system on the data input line 34 is supplied to a first internal processor register 50 and an output signal from a

second internal processor register 52 is supplied on the data strobe line 30 to the test system. These two lines afford an interactive, two-way communication link between the processor and the test system in that the test system may run corresponding software which works in conjunction with the controller code running on the processor to allow the test system to pass command packets to the processor via the test code input line, and the processor to send status in the form of packets or specific data patterns to the test system via the data strobe line. When passing data from the processor to the test system via the data strobe line or when passing data from the test system to the processor via the data input line, the data transmission speed is not coupled to the processor's operating speed, but rather, it is controlled by the software running on both the test system and the processor.

Detailed Description Text (17):
Through use of the test code input 44 and the MUX 36, the test system can send command packets to the processor. The controller code may parse the command packets sent from the test system and then execute the command received at the processor's operating speed. Through a command such as "load", module resident memory/storage components or registers may be loaded with data sent from the test system, and through a "dump" command, these same locations may be examined. A "start" command may be used to initiate a previously loaded test or program. In this way, the test system may direct the processor to initiate certain tests, step through certain tests, load specific locations with data, read specific locations, etc.

Detailed Description Text (19):
These are only some of the possible commands that the test system can send to the processor.

Detailed Description Text (20):
Once the two-way communication link is established, the test system 22 does not have to wait for the processor to request data as was the case during initialization. Instead, the test system can send commands packets interactively via the communication link. The test system can send command packets at its discretion serially through the data input line 34 by packaging them appropriately with start and stop bits. When the hardware detects the start bit, the controller code may be notified and the packet may be accepted. The controller code running on the processor may be notified that a start bit has been received by having an interrupt generated when the internal register bit connected to the data line transitions between different logic states indicating that a start bit has been received, or the controller code may detect a transition of the register bit itself by polling the register bit to detect the reception of a start bit. The controller code may then parse the packet and, from information gained through the parse, determine the command received and the amount of data which constitutes the packet. The controller code will look for a stop bit when the expected amount of data has been received. Checksums or other error checking means can also be included in the packet to permit determination of the integrity of the received packet. The controller code will then execute the command received at the processor's operating speed and report status via the data strobe line 30, if so requested by the test system.

Detailed Description Text (21):
Through the two-way communication link, a packet may be used to set the controller code up for a burst mode transfer to enable comprehensive testing. A burst mode transfer will allow comprehensive tests which are expressed in more code than may be transferred to the processor during initialization due to the limitation imposed by the size of the processor's I-cache to be transferred to memory/storage components on the MUT 20 external to the processor 24 such as external cache 54 shown in FIG. 2. Many other types of memory components can be loaded with code in this way, for instance, flash PROMs (programmable ROMs), RAM (random access memory), and EEROMs (electrically erasable ROMs). After determining the defect on a

MUT, a user such as a field service representative could load memory components with possible fixes for that defect. As will be described shortly, prior to a burst mode transfer, tests loaded into the processor may be executed by the controller code to determine if such external memory components are working properly.

Detailed Description Text (22):
The packet containing the burst mode command will provide an address indicating where the data received should be stored on the MUT as well as the amount of data to be sent. The controller code can then use the data strobe line 30 to request the data as was done during initialization without having to re-initialize the processor. This will allow a burst mode transfer to take place at a speed dictated by the speed of the controller code running on the processor and the corresponding software running on the test system. The controller code can use the data strobe line to report the status of the packet following the burst mode transfer if requested to do so by the test system.

Detailed Description Text (23):
The test system may include a user interface such as a video terminal 56 connected to a test computer 58. Test computer 58 is connected to the MUT 20 through the cable 40 and may serve as a means for controlling communications between the video terminal 56 and the MUT. The test computer may run software that works with controller code running on the processor such that user commands, entered through a keyboard on the video terminal 56, can be translated into user command packets and sent to the processor, via the test code input line 44, to be received and parsed by the controller code. The software running on the test computer 58 may request that the controller code return status after executing the commands sent. The test computer 58 may have means for receiving status via the processor's data strobe line 30 and means for translating this status into appropriate responses which are sent to the video terminal 56.

Detailed Description Text (24):
The software running on test computer 58 can load a set of pre-written tests into the processor on initialization of the module, or controller code can be loaded followed by an interactive loading of pre-written test code. If a failure occurs, whether testing is being conducted in volume manufacturing, during initial module testing, or by field service representatives at customer sites, a user involved with testing the MUT can use the video terminal 56 to select pre-written tests to load interactively into the processor or to step through a particular test or test sequence. A user familiar with programming may choose to write new tests to load interactively. The test system 22 can then be used to probe the MUT 20 through the processor 24 without losing error information due to re-initialization. The user can use the test system to determine more information about the failure in an attempt to specifically locate the defect.

Detailed Description Text (25):
The testing sequence loaded is completely at the user's discretion. Because the processor 24 is loaded with test code from the MUX 36 to which it is directly connected, only a small amount of untested hardware is relied upon to load the processor. This allows module testing using an "ever-increasing sphere approach," meaning that the testing sequence can begin with testing the processor 24 itself and then gradually move outward from the processor to progressively test other components on the MUT 20 and the computer system to which the MUT is connected. Increasing the sphere of hardware being tested by small increments of untested hardware allows an error to be isolated to a defect in the new hardware being tested. The two-way communication link allows the user to write and load tests to probe the area where the defect is believed to be without having to re-initialize the processor and risk losing an intermittent error.

Detailed Description Text (26):
The invention provides a relatively non-intrusive, processor-based computer module

testing method and apparatus which allows the user maximum flexibility in testing the MUT 20 and the computer system for which the MUT was designed to be a part of.

US Reference Patent Number (10):
4811345

CLAIMS:

1. An apparatus, comprising:

a computer module to be tested, including:

a processor having a data input path for accepting code;

means for storing initialization code, wherein the initialization code is executed by the processor to initialize the operation of the processor; and

interrupting means connected to the storing means and to the processor data input path and disposed between the storing means and the processor; and

means, coupled to the interrupting means, for inputting test code to the processor, wherein the interrupting means are responsive to the inputting means and the inputting means are capable of causing the interrupting means to substitute the test code for the initialization code to the processor on the processor data input path.

2. An apparatus according to claim 1, wherein the interrupting means comprises an element having a first input for the initialization code, a second input for the test code, an output connected to the processor data input path, and means for controlling the element to selectively connect the output to the first and second inputs.

7. An apparatus according to claim 1, wherein the inputting means comprises a test system and communication means disposed between the test system and the interrupting means for supplying the test code to the processor.

15. An apparatus according to claim 1, wherein the test code comprises processor initialization code and a testing sequence to be executed by the processor to test the computer module.

16. An apparatus according to claim 15, wherein the computer module further comprises a plurality of components which interface the computer module to a computer system, and wherein the testing sequence to be executed by the processor is formed to test the processor and progressively to test other components on the computer module.

19. A method of testing a computer module which includes a processor which receives code for initializing the operation of the processor, comprising the steps of:

interrupting an initialization path to the processor; and

inputting test code to the processor for execution by the processor as a substitute for the code received by the processor during initialization to initialize the processor and initiate testing of the computer module.

23. A method of testing a computer module according to claim 22, wherein the step of inputting the test code to the processor comprises the steps of:

asserting a reset line to the processor;

deasserting a reset line to the processor;

receiving a data strobe line from the processor; and

providing test code to the second data input of the multiplexor.

24. A method of testing a according to claim 19, wherein the step of inputting test code to the processor comprises inputting test code at a speed different from an operating speed of the processor.

25. A method of testing a computer module according to claim 19, wherein the computer module is part of a computer system, and wherein the testing includes testing the computer system at the processor's operating speed.

L10: Entry 7 of 10 File: USPT Oct 11, 1994

DOCUMENT-IDENTIFIER: US 5355369 A
TITLE: High-speed integrated circuit testing with JTAG

Brief Summary Text (3):
The present invention relates to an integrated circuit (IC) that implements the JTAG test port while providing for the high-speed testing of a programmable digital processor.

Brief Summary Text (12):
We have invented a technique for testing an integrated circuit having a digital processor. The integrated circuit incorporates circuitry that allows for testing the digital processor at full operational speed. To perform the test, the only input/output signal pins required are those carrying the JTAG TAP signals, and optionally a system clock, allowing the integrated circuit to be mounted in a board during testing. Means for block downloading of a test program may optionally be included.

Detailed Description Text (2):
The following detailed description relates to an integrated circuit having a digital processor, and incorporating circuitry that allows for testing the digital processor at full speed. The digital processor is typically a digital signal processor (DSP) or microcomputer, wherein an arithmetic logic unit (ALU) resides on the same integrated circuit chip as the program memory. The integrated circuit may be mounted in a board during testing, and the only signal pins required are those carrying the JTAG TAP signals, with a high speed operational clock signal (i.e., the system clock) being optional. Two additional registers that are allowed under the "optional" clause of the JTAG standard have been defined, and are referred to as JCON and TDR herein. Both of these registers may be selected by sending the proper serial sequence to the TAP controller. Referring to FIG. 1, both JCON and TDR appear to the JTAG master as serial scan registers (10 and 11, respectively). JCON has parallel output on the chip. Six of its bits are used for the down-loadable self-test; they are:

Detailed Description Text (6):
Also included in the illustrative embodiment is a down-loadable digital processor architecture (see FIG. 2). The illustrative digital processor utilizes "harvard" architecture, which has separate address and data buses for "Instruction/Coefficient" and "Data". These buses communicate with the ALU (21), the arithmetic address unit AAU (23), and the controller (24). It also has dual port RAM (22) which communicates with both sets of buses. A program may be downloaded to the chip by having the digital processor core read an instruction as "Data" via one of its many data input mechanisms and writing it as "Data" to the dual port RAM. The instruction may then be executed from fetches over the "Instructions/Coefficient" buses from the dual port RAM. The scheme used in the illustrative processor uses a dual port memory associated with the digital processor core and having a size of typically at least 256 words. Note that unlike the BIST testing method, the memory requirement of the inventive testing technique does not necessarily add to chip area. This is because the memory may be re-used after testing for digital processor application programs and data. Although a dual-port memory is convenient for implementing the present invention, its use is not

mandatory. For example, a Von Neumann architecture allows the use of a single-port memory. Note that the multiplexer 25, under control of signals CKTCK and SELCKI, provides a clock from any one of TCK (the JTAG test clock), CKI (the system clock used in normal chip operation), or JCKI (the scanned system clock).

Detailed Description Text (7):
A read-only memory "JROM" (20) resides in the digital processor core to control the downloading of test programs into the dual port RAM. This memory has six words in the illustrative case, and the assembly language program in the JROM is:

Detailed Description Text (10):
In order to execute the downloaded program, the JTAG master resets the JLOAD signal in JCON. This removes the JROM from the instruction space and maps the instruction port of the dual port RAM back in. The JTAG master sets the JPRLOW bit to select a program space memory map where the dual port RAM starts at location 0. It pulses the JRESET signal in JCON to reset the digital processor core, causing the downloaded test program to begin. Each test program produces results that are sent back to the JTAG master. The digital processor test program writes a result to the TDR. This clears the PINT signal. The JTAG master selects the instruction register and polls the PINT signal until it sees that a result has been written. The digital processor program does a conditional branch on the flag that is tied to PINT, looping until it sees PINT go high. When the JTAG master shifts out the result, PINT goes high and the digital processor program may write the next result. Before downloading a test, the JTAG master scans a desired condition into the Boundary Scan register. By using the JTAG Instruction Codes in the above table, the output and bidirectional pins of the chip will be held in the state of the Boundary Scan register during the test.

Detailed Description Text (11):
The JCON register optionally contains two additional signals to provide flexibility in clocking the digital processor test program while it is in a board environment. The board containing the digital processor normally provides a system clock to the CKI terminal. If this clock is available and free running on the board when the test is to be run, the JTAG master may select it as the source of chip clocking by setting the SELCKI signal in JCON, which allows the test to be run in situ at full speed. If the clock on the board is not running, it is still possible to clock the chip by two other methods. The default is the standard JTAG method, where the CKI may be scanned into the Boundary Scan register. This method is extremely slow. The other option is for the JTAG master to set the CKTCK signal in JCON. This causes the digital processor to be clocked by the JTAG test clock, TCK, supplied by the JTAG master. The speed of this clock is dependent on the system implementation.

Detailed Description Text (18):
Upon completion of the processing of the downloaded test program by the digital processor, the results are then uploaded to the TDR (50) from the data bus 54. This may be accomplished by directly writing the result to the TDR (a parallel bit transfer) by the digital processor, since the TDR is directly addressable therefrom. The result may then be serially scanned out via the JTAG output port TDO, according to the standard JTAG technique. In the illustrative arrangement of FIG. 4, the output passes through the bypass registers (408, 410, 412) of the chips (407, 409, 411) following the target chip. Note that the result of the test is typically several words, which require several such transfers. However, it is alternatively possible to compress the resulting words into fewer words, or even only one word, for transfer through the TDO port. Although an integrated circuit that implements the present invention may also implement all of the standard JTAG functions, that is not necessary in all cases. For example, the boundary-scan register may be omitted, while still advantageously implementing the present test using the TAP controller, instruction decoder, and TDI, TDO, TMS, and TCK pins on the integrated circuit. The prior-art use of user test registers (306) for implementing BIST may also be included in IC's that implement the present

invention. Still other variations are possible, and included herein.

US Reference Patent Number (3):
4811345

CLAIMS:

1. An integrated circuit having input/output ports and comprising a programmable digital processor connected to a program memory having an address space;

and further comprising means for performing boundary-scan testing on the input/output ports of said integrated circuit, wherein said means comprises a finite state machine controller connected to an instruction register that is connected to an instruction decoder, a serial test input port, and a serial test output port coupled to said instruction register;

wherein said integrated circuit further comprises:

a test data register (TDR) connected to said finite state machine controller and said instruction decoder, and having n-bit locations for serially receiving a test program through said serial test input port, and for transferring said test program in parallel n-bit words into said program memory via said digital processor;

and a test control register (JCON) connected to said finite state machine and said instruction decoder for initiating downloading and execution of said test program in order to produce test results;

and wherein the test results of said test program are transferred via said programmable digital processor into said test data register for serial transfer through said serial test output port.

4. The integrated circuit of claim 1 wherein said test control register includes a control bit for clocking said digital processor from a signal supplied from a test clock external to said integrated circuit in lieu of a clock supplied from said means for performing boundary scan testing.

13. An electronic system comprising a multiplicity of integrated circuits that implement a boundary-scan test according to a Joint Test Action Group (JTAG) boundary-scan test standard, wherein a given integrated circuit comprises a programmable digital processor connected to a program memory having an address space;

and wherein said given integrated circuit further comprises:

a test data register (TDR) controlled by a JTAG Test Access Port (TAP) controller and a JTAG instruction decoder, and having n-bit locations for serially receiving a test program through a Test Data Input (TDI) serial test input port, and for transferring said test program in parallel n-bit words into said program memory via said programmable digital processor;

and a test control register (JCON) controlled by said TAP controller and said JTAG instruction decoder for initiating downloading and execution of said test program in order to produce test results;

and wherein the test results of said test program are transferred via said programmable digital processor into said test data register for serial transfer through a Test Data Output (TDO) serial test output port.

16. The system of claim 13 wherein said test control register includes a control bit (JRESET) for resetting the programmable digital processor and beginning test

program execution at a given memory location.

21. An integrated circuit comprising a programmable digital processor connected to a program memory;

and further comprising a finite state machine controller connected to an instruction register that is connected to an instruction decoder, a serial test input port and a serial test output port coupled to said instruction register;

wherein said integrated circuit further comprises:

a test data register (TDR) connected to said finite state machine controller and said instruction decoder, and having n-bit locations for serially receiving a test program through said serial test input port, and for transferring said test program in parallel n-bit words into said program memory;

and a test control register (JCON) controlled by said finite state machine controller and said instruction decoder for initiating downloading and execution of said test program in order to produce test results;

and wherein said programmable digital processor uploads the test results of said test program into said test data register for serial transfer through said serial test output port,

and further comprising means for enabling block downloading of said test program into said test data register, wherein said test program is received as a continuous sequence of data words after receiving an initial sequence of a given bit pattern, and wherein said given bit pattern that is received is a final "1" preceded by a series of all zeros.

L10: Entry 10 of 10 File: USPT Jul 28, 1992

DOCUMENT-IDENTIFIER: US 5134701 A
TITLE: Test apparatus performing runtime replacement of program instructions with breakpoint instructions for processor having multiple instruction fetch capabilities

Abstract Text (1):
The test apparatus for monitoring the operation of a processor that has multiple instruction fetch capability monitors the instruction memory to record the sequence of program instructions that are retrieved by the processor from program memory. The test apparatus determines when a jump operation is executed and determines the target of the jump oepration by inserting a break point instruction in place of one of the two program instructions that is retrieved by the processor from program memory. This instruction substitution is accomplished by an instruction jamming circuit that forces the break point instruction onto the processor data bus as part of the program instruction fetch cycle in lieu of one of the instruction retrieved as part of the execution of the jump instruction. If the break point operation is executed, then the target address of the jump operation is the address location that contains the break point instruction that was substituted for one of the program instructions retrieved from the instruction memory. In this case, the test apparatus responds to the execution of the break point instruction by replacing the program instruction originally retrieved from program memory and substituted for by the break point instruction. Thus, the break point instruction acts as a flag to indicate that this address is the target address of the jump instruction. If the break point instruction is not executed by the processor, it is because the jump instruction target address is the location that contains the other retrieved program instruction.

Brief Summary Text (2):
This invention relates to test apparatus for monitoring the operation of a processor, and in particular, apparatus for tracing the step-by-step program execution of a processor that has the capability to retrieve a plurality of program instructions in a single instruction fetch.

Brief Summary Text (6):
The above described problems are solved and a technical advance achieved in the field by a test apparatus for monitoring the operation of a processor that has multiple instruction fetch capability. This test apparatus monitors the instruction fetches from program memory. The test apparatus detects when the processor executes a jump operation and determines the target address of the jump operation by inserting a break point instruction in place of one of the program instructions that are retrieved by the processor from program memory. This instruction substitution is accomplished by an instruction jamming circuit that forces the break point instruction onto the processor data bus as part of the program instruction fetch cycle in lieu of one of the instructions retrieved as part of the instruction fetch after the execution of the jump instruction. If the break point operation is executed, then the target address of the jump operation is the address location that contains the breakpoint instruction that was substituted for one of the program instructions retrieved from the instruction memory. In this case, the

test apparatus responds to the execution of the break point instruction by replacing the program instruction originally retrieved from program memory and substituted for by the break point instruction. Thus, the break point instruction acts as a flag to indicate, when executed, that this address is the target address of the jump instruction. If the break point instruction is not executed by the processor, it is because the jump instruction target address is the location that contains the other retrieved program instruction. Thus, the execution or non-execution of the break point instruction acts as a flag to indicate to the test apparatus which of the retrieved program instructions that the processor has fetched as a result of the jump instruction was actually executed. In this manner, the test apparatus provides an accurate record of the sequence of program instructions that are actually executed by the processor under test.

Drawing Description Text (2):
FIG. 1 illustrates, in block diagram form, the test apparatus for monitoring the operation of a processor that has the capability of retrieving a plurality of program instructions;

Drawing Description Text (4):
FIG. 3 illustrates, in flow diagram form, the operation of the test apparatus for monitoring the operation of a processor that has the capability of retrieving a plurality of program instructions; and

Detailed Description Text (4):
FIG. 1 illustrates in block diagram form the test apparatus for monitoring the operation of a processor that can retrieve a plurality of program instructions during each instruction fetch cycle and the interconnection of this test apparatus with a system under test. The test apparatus 100 is interconnected with a processor controlled system under test 101 via a set of cables 102. The interconnection of the test apparatus 100 and the system under test 101 is such that the processor status (121), address (120) and data busses (122) in the system under test 101 are interconnected with the test apparatus 100. This interconnection enables test apparatus 100 to monitor the exact operation of the system under test 101.

Detailed Description Text (5):
Test apparatus 100 typically includes a host such as a computer 109 that is equipped with a keyboard 108 and a video monitor 110. The function of computer 109 and the additional circuitry (not shown) found in test apparatus 100 is to record the sequence of program instruction operations executed by the processor 111 in system under test 101. The function of this additional circuitry in test apparatus 100, as well as the software in computer 109, is well known and is not disclosed in detail herein. Computer 109 can be for example a Hewlett Packard Model 9000, Series 300 computer and the additional test apparatus is included in for example a Hewlett Packard 64120 instrumentation package.

Detailed Description Text (6):
The recording and analysis circuitry and software of test apparatus 100 typically cannot exactly follow the instruction flow of the processor 111 in system under test 101 since the program flow can become nonsequential whenever a jump, branch, interrupt or other similar operation is executed by the processor 111. In these situations, the processor 111 may fetch multiple instructions from a new address, called the target address, rather than following the sequence of operations that were originally retrieved from program memory. While it is possible to either replicate or simulate the internal operation of the processor 111 in system under test 101 and thereby uncover which instructions were and were not executed, this requires extensive human intervention and is expensive and subject to error. Emulator pod 103 functions to indirectly force the processor 111 in system under test 101 to reveal which instructions were not executed. This is accomplished without the execution of a software monitor or the need to execute a dump operation which significantly impacts on the real time operation of the system under test

101.

Detailed Description Text (7):
The interconnection illustrated in FIG. 4 represents the emulation of the operation of the processor 111 in the system under test 101. In this application, the processor 111 normally found in the system under test 101 is replaced by the emulator pod 103 which functions in the same manner as the processor 111 removed from the system under test 101. Cable 102 plugs into the connector in which the processor 111 in system under test 101 is normally installed. Cable 102 brings the address 120, status 121 and data 122 busses from system under test 101 to a processor 104 in emulator pod 103 as illustrated in FIG. 1. Processor 104 is either the same device as processor 111 or is a device that can emulate the operation of processor 111 so that processor 104 executes the program instructions in system under test 101 in transparent fashion. The processor address 120 and status 121 busses of system under test 101 are connected directly to processor 104. The processor data bus 122 of system under test 101 is connected to processor 104 by way of buffer 105, the function of which is described below. Thus, emulator pod 103 functions as processor 111.

US Reference Patent Number (15):
4811345

CLAIMS:

1. In a test system that monitors the operation of a system under test, which system under test includes a processor that retrieves a plurality of program instructions from a program memory during a program instruction fetch cycle for immediate execution of said retrieved program instructions by said processor, wherein each program instruction is stored in a designated location in said program memory, each said designated location having a memory address, a program instruction trace apparatus for determining a sequence of program instructions executed by said processor comprising:

means, responsive to said processor retrieving a plurality of program instructions from said program memory for immediate execution of the retrieved program instructions by said processor, for comparing the memory address of each of said plurality of program instructions retrieved by said processor during a presently executing program instruction fetch cycle with the memory address of the last previously retrieved program instruction to determine whether said presently retrieved program instructions represent an out of sequence program instruction fetch;

means, responsive to said comparing means, for substituting a predefined program instruction for one of said presently retrieved program instructions when said presently retrieved program instructions represent an out of sequence program instruction fetch; means for storing said one presently retrieved program instruction; and

means, responsive to said processor executing said predefined program instruction, for replacing said executed predefined program instruction with said one retrieved program instruction.

8. In a test system that monitors the operation of a system under test, which system under test includes a processor that retrieves a plurality of program instructions from a program memory during a program instruction fetch cycle, wherein each program instruction is stored in a designated location in said program memory, each said designated location having a memory address, a program instruction trace apparatus for determining a sequence of program instructions executed by said processor comprising:

means for comparing the memory address of each of said plurality of program instructions retrieved by said processor during a presently executing program instruction fetch cycle with the memory address of the last previously retrieved program instruction to determine whether said presently retrieved program instructions represent an out of sequence program instruction fetch;

means, responsive to said comparing means, for substituting a predefined program instruction for one of said presently retrieved program instructions when said presently retrieved program instructions represent an out of sequence program instruction fetch;

means, responsive to said processor executing said predefined program instruction, for replacing said executed predefined program instruction with said one retrieved program instruction;

means for recording said retrieved program instructions; and

means for deleting one or more of said recorded program instructions in said plurality of program instructions retrieved in said one program instruction fetch cycle when said processor fails to execute said predefined program instruction.

13. In a test system that monitors the operation of a system under test, which system under test includes a processor that retrieves a plurality of program instructions from a program memory during a program instruction fetch cycle for immediate execution of said retrieved program instructions by said processor, wherein each program instruction is stored in a designated location in said program memory, said designated location having a memory address, a method of determining a sequence of program instructions executed by said processor comprising the computer implemented steps of:

comparing, in response to said processor retrieving a plurality of program instructions from said program memory for immediate execution of the retrieved program instructions by said processor, the memory address of each of said plurality of program instructions retrieved by said processor during a presently executing program instruction fetch cycle with the memory addresses of the last previously retrieved program instructions to determine whether said presently retrieved program instructions represent an out of sequence program instruction fetch;

substituting a predefined program instruction for one of said presently retrieved program instructions when said presently retrieved program instructions represent an out of sequence program instruction fetch; storing said one presently retrieved program instruction; and

replacing, in response to said processor executing said predefined program instruction, said executed predefined program instruction with said one retrieved program instruction.

19. In a test system that monitors the operation of a system under test, which system under test includes a processor that retrieves a plurality of program instructions from a program memory during a program instruction fetch cycle, wherein each program instruction is stored in a designated location in said program memory, said designated location having a memory address, a method of determining a sequence of program instructions executed by said processor comprising the computer implemented steps of:

comparing the memory address of each of said plurality of program instructions retrieved by said processor during a presently executing program instruction fetch cycle with the memory addresses of the last previously retrieved program instructions to determine whether said presently retrieved program instructions

represent an out of sequence program instruction fetch;

substituting a predefined program instruction for one of said presently retrieved program instructions when said presently retrieved program instructions represent an out of sequence program instruction fetch;

replacing, in response to said processor executing said predefined program instruction, said executed predefined program instruction with said one retrieved program instruction;

recording said retrieved program instructions; and

deleting one or more of said recorded program instructions in said plurality of program instructions retrieved in said one program instruction fetch cycle when said processor fails to execute said predefined program instruction.

23. In a test system that monitors the operation of a system under test, which system under test includes a processor that retrieves a plurality of program instructions from a program memory during each program instruction fetch cycle for immediate execution of said retrieved program instructions by said processor, wherein each program instruction is stored in a designated location in said program memory, said designated location having a memory address, a program instruction trace apparatus for determining a sequence of program instructions executed by said processor comprising:

means, responsive to said processor retrieving a plurality of program instructions from said program memory for immediate execution of the retrieved program instructions by said processor, for intercepting one of said plurality of program instructions retrieved by said processor during a program instruction fetch cycle in response to an out of sequence program instruction fetch;

means, responsive to said intercepting means, for substituting a predefined program instruction for said one intercepted program instruction; means for storing said one intercepted program instruction; and

means, responsive to said processor executing said predefined program instruction, for replacing said executed predefined program instruction with said one intercepted program instruction.

29. In a test system that monitors the operation of a system under test, which system under test includes a processor that retrieves a plurality of program instructions from a program memory during a program instruction fetch cycle, wherein each program instruction is stored in a designated location in said program memory, said designated location having a memory address, a program instruction trace apparatus for determining a sequence of program instructions executed by said processor comprising:

means for comparing the memory address of each of said plurality of program instructions retrieved by said processor during a presently executing program instruction fetch cycle with the memory address of the last previously retrieved program instruction to determine whether said presently retrieved program instructions represent an out of sequence program instruction fetch;

means, responsive to said comparing means, for substituting a predefined program instruction for one of said presently retrieved program instructions when said presently retrieved program instructions represent an out of sequence program instruction fetch, including:

means, interposed between said processor and said program memory, for transmitting said predefined program instruction to said processor in lieu of said one retrieved

program instruction,

buffer means for intercepting said one retrieved program instruction,

means, responsive to said processor executing said predefined program instruction, for replacing said one retrieved program instruction, including:

means, responsive to said processor executing said predefined program instruction, for reading said intercepted retrieved program instruction from said buffer means to said processor; means for recording said retrieved program instructions;

means for deleting said recorded program instructions in said plurality of program instructions retrieved in said one program instruction fetch cycle when said processor fails to execute said predefined program instruction.

30. In a test system that monitors the operation of a system under test, which system under test includes a processor that retrieves a plurality of program instructions from a program memory during each program instruction fetch cycle, wherein each program instruction is stored in a designated location in said program memory, said designated location having a memory address, a program instruction trace apparatus for determining a sequence of program instructions executed by said processor comprising:

means for intercepting one of said plurality of program instructions retrieved by said processor during a program instruction fetch cycle in response to an out of sequence program instruction fetch;

means, responsive to said intercepting means, for substituting a predefined program instruction for said one intercepted program instruction;

means, responsive to said processor executing said predefined program instruction, for replacing said executed predefined program instruction with said one intercepted program instruction;

means for recording said retrieving program instructions; and

means for deleting one or more of said recorded program instructions in said plurality of program instructions retrieved in said one program instruction fetch cycle when said processor fails to execute said predefined program instruction.

Refine Search

Search Results -

Terms	Documents
L2 same test$	37

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L3

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Thursday, June 17, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L3	L2 same test$	37	L3
L2	motherboard same interface same host	416	L2
L1	motherboard same tester same interface	16	L1

END OF SEARCH HISTORY

L3: Entry 34 of 37 File: USPT Feb 18, 1997

DOCUMENT-IDENTIFIER: US 5604888 A
** See image for Certificate of Correction **
TITLE: Emulation system employing motherboard and flexible daughterboards

Detailed Description Text (12):
The controller chip 26 for the motherboard 10 is connected to the emulation bus 24 to aid in the configuration loading of the chips on the daughter cards 4, 6 and 8, and for the debugging and fault testing of the emulated circuit. The controller chip 26 forms the interface with the host computer 40. The controller chip 26 can be an FPGA that is configured upon boot-up by a programmable read-only memory (PROM) 42 that stores the configuration data of the controller chip 26. Upon boot-up, the configuration data for other controller chips on the daughter cards or motherboard can be loaded through the controller chip 26 to these other controller chips such as controller chips 32 and 36 or controller chips (not shown) for the crosspoint array or tester interface.

Detailed Description Text (19):
In step 62, the routing of the emulation circuit through the crosspoint array 22 including the assigning of different pins of the emulation field programmable gate arrays is done. In step 64, the field programmable gate array configuration data for the circuit-emulation FPGAs is compiled and loaded through the motherboard and daughterboard controller chips. The field programmable gate arrays have their own compiler software that can be stored in the host computer 40 so that the FPGA configuration data can be produced. The FPGA compiler software can be obtained from the FPGA vendor or from independent sources. In step 66, the FPGA configuration data for the crosspoint arrays FPGA's is compiled and loaded from the motherboard. At this point, the system is ready for the emulation to be run in step 68. In step 70, debugging is now possible. The state of the nodes in the emulation can be determined through the daughterboard chip controllers. During the debugging phase, the system provides the capacity to probe internal nodes of the design. The temporary logic needed to probe local circuits is automatically added and removed via graphical user interface. This temporary logic can be stored in the tester interface 38. The system allows for test vector verification capacity and allows the user to download their test vectors into the system and compare these results against the simulation results. This system can also provide an interface to the user's existing software simulator, or vary the compiled design prior to emulation. This gives the capacity to simulate the design complete with fully back-annotated timing information. This system can also provide an interface to a user's existing timing analyzer. This gives the capacity to analyze the partition design for hold violations, and predict the maximum speed of the emulated design. The existing pad system provides more accurate results using industry standard and mature products. The entry-level emulation is much cheaper with this system, as the system works within the existing computer-aided design framework rather than replacing it.

L1: Entry 5 of 16 File: USPT Feb 19, 2002

DOCUMENT-IDENTIFIER: US 6348810 B1
TITLE: Interface unit for a tester and method of connecting a tester with a semiconductor device to be tested

Brief Summary Text (2):
The present invention relates generally to interface units for testers that test electrical characteristics of IC (Integrated Circuit) devices and other semiconductor devices. More particularly, the present invention relates to a tester interface unit for providing electrical connection between a contact ring and a motherboard, and a method of connecting the tester with a semiconductor device to be tested.

Brief Summary Text (8):
In order to accomplish the above-mentioned object, the present invention provides an interface unit for a tester which comprises: a contact board having a large number of contact terminals provided on one surface thereof, and sockets provided on another surface thereof and electrically connected with the contact terminals; a motherboard having a large number of wires to be connected to the contact terminals of the contact board, and a large number of sockets electrically connected with the wires; and a large number of connecting cables each having two plugs provided at opposite ends thereof. The plug provided at one end of each of the connecting cables is removably inserted in one of the sockets of the contact board and the plug provided at another end of each of the connecting cables is removably inserted in one of the sockets of the motherboard, so that the connecting cables electrically connect the contact terminals of the contact board with the wires of the motherboard.

Brief Summary Text (9):
The tester interface unit is used for providing electrical connection between a test control section and a probe section of a tester for an IC or other semiconductor device. The wires of the motherboard are connected to the test control section and the contact terminals provided on the one surface of the contact board are placed in contact with contact terminals of the probe section. The probe section is a probe card having the above-mentioned contact terminals on one surface thereof and contact needles, such as tungsten needles, on the other surface thereof, and the contact terminals of this probe section are electrically connected with the contact needles. When performing a test, the contact needles of the probe section are placed in contact with predetermined terminals of the device to be tested such as a semiconductor wafer, semiconductor device or IC.

Drawing Description Text (3):
FIG. 1 is a perspective view showing specific constructions of a motherboard, coaxial cables and contact ring in a tester interface unit in accordance with a preferred embodiment of the present invention;

Detailed Description Text (12):
Furthermore, whereas the present invention has been described above in relation to a probe tester for testing IC devices in the form of a semiconductor wafer, it should be apparent that the present invention is also applicable as an interface unit for providing electrical connection between the contact ring and the

motherboard. Moreover, whereas the present invention has been described in relation to an IC tester, it can of course be applied to a package tester, logic tester and the like. Therefore, the contact ring employed in the present invention does not have to be always ring-shaped and may be a plate having any desired shape.

CLAIMS:

7. An interface unit as recited in claim 1 which is used for providing electrical connection between a test control section and a probe section of a semiconductor tester, and wherein the conductors of said motherboard are connected to the test control section and the contact terminals provided on the one surface of said contact board are placed in contact with contact terminals of the probe section.

L1: Entry 8 of 16 File: USPT Nov 16, 1999

DOCUMENT-IDENTIFIER: US 5986447 A
TITLE: Test head structure for integrated circuit tester

Abstract Text (1):
A test head for an integrated circuit tester includes a horizontal base holding a circular motherboard. The motherboard distributes input test instructions to an array of carrier boards mounted thereon, the carrier boards being radially distributed about a central vertical axis of the motherboard. Each carrier board holds a set of daughterboards, and each daughterboard holds a set of node cards. The carrier boards and daughterboards include data paths for forwarding the test instructions from the motherboard to the node cards. Each node card contains circuits for transmitting test signals to and receiving response signals from a separate terminal of a device under test (DUT) in response to the test instructions forwarded thereto. Edges of the carrier boards extend downward through apertures in the base to contact pads on an interface board holding the DUT. The carrier boards and daughterboards provide conductive paths for the test and response signals extending between the node cards and pads on the DUT interface board. The interface board extends those conductive paths from the pads to terminals of the DUT.

Brief Summary Text (15):
A test head for an integrated circuit tester in accordance with the present invention includes a horizontal base holding a printed circuit "grandmother" board. The motherboard distributes input test instructions to an array of carrier boards mounted thereon, the carrier boards being radially distributed about a central vertical axis of the motherboard. Each carrier board holds a set of daughterboards, and each daughterboard holds a set of "node" cards. The carrier boards and daughterboards include data paths for forwarding the test instructions from the motherboard to the node cards. Each node card contains circuits for transmitting test signals to and receiving response signals from a separate terminal of a device under test (DUT) in response to the test instructions forwarded thereto. Edges of the carrier boards extend downward through apertures in the base to contact pads on an interface board holding the DUT. The carrier boards and daughterboards provide conductive paths for the test and response signals extending between the node cards and pads on the DUT interface board. The interface board extends those conductive paths from the pads to terminals of the DUT.

Detailed Description Text (7):
FIGS. 3-9 illustrate the physical structure of test head 24 for integrated circuit tester 10 in accordance with the present invention. Referring to FIGS. 3-9, test head 24 includes a horizontally disposed interface board 40 for holding a device under test (DUT) 11 in a socket (or probe head) 42. A "grandmother" board 44 is mounted on a base plate 46 positioned immediately above and parallel to DUT interface board 40. Lower edges of a set of carrier boards 50 are inserted into connectors 52 mounted on an upper surface 54 of motherboard 44. A cylindrical support member 56, attached to base plate 46 and extending vertically upward from motherboard 44, includes a set of card holders 58 distributed around its outer surface 60 each for receiving a front edge 62 of a separate carrier board 50. Connectors 52 and holders 58 hold carrier boards 50 in vertical planes perpendicular to the horizontal plane of surface 54 of motherboard 44 with the plane of each carrier board 50 bisecting the vertical axis 64 of cylindrical

support member 56.

L1: Entry 13 of 16 File: USPT Oct 17, 1995

DOCUMENT-IDENTIFIER: US 5459738 A
TITLE: Apparatus and method for digital circuit testing

Detailed Description Text (2):
A digital circuit tester 40 of an embodiment of the invention is schematically shown in FIG. 2 connected via an interface 42 to a processor 44. The processor 44 for controlling and operating the digital circuit tester may be within a personal computer, for example. An interface with the processor may be formed through the RS-232 serial port, the parallel printer port or as presently preferred by direct connection of the digital circuit tester into the motherboard of the personal computer. Other conventional means of interfacing with a processor may also be used by a digital circuit tester within the scope of the invention. As is known, the processor 44 is in communication with memory 46. A test for a digital circuit includes a sequence of node drive commands and an expected output pattern. Tests and their accompanying output patterns are known for many of the available integrated circuits on the market. There are many acceptable and conventional methods for making alternative tests for existing circuits as well as for any new integrated circuit introduced on the market.

Detailed Description Text (12):
Referring now to FIG. 5, the tester of one embodiment of the invention is shown mounted on a printed circuit board that is insertable into the motherboard of a personal computer. The tester can then interface directly with the ISA bus of the PC. The connections between the tester and the circuit to be tested are conducted through a flat cable 60. The cable should preferably be kept short to avoid problems associated with transmission delays. The flat cable may be connected to a pod 62. The pod 62 may be provided with an AC termination to prevent reflections of the applied signals. The pod 62 may include a ZIF socket 64. The ZIF socket is used for insertion of a DIP clip 58. The DIP clip 58 is used to connect the tester to a circuit to be tested on a circuit board. The pod may be used to permit a circuit to be tested out-of-circuit by insertion directly into the ZIF socket. Thus, the tester may be used for in-circuit or out-of-circuit testing. The tester of the invention provides service technicians and manufacturers with a low cost alternative for testing digital circuits.

CLAIMS:

17. The circuit tester of claim 16 wherein said interface comprises an electrical connector for insertion into a motherboard of a computer.

20. The digital circuit tester of claim 19 wherein said interface comprises an electrical connector for insertion into a motherboard of a computer.